Business Case : Yulu - Hypothesis Testing

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Importing Libraries and Data

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df=pd.read_csv('yulu.csv')
df.head(5)
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

1.Define Problem Statement and perform Exploratory Data Analysis

1.1 Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

Shape of data

In [ ]:
```
df.shape
```

Out[ ]:
```
(10886, 12)
```

In [ ]:
```
print(f'Number of rows = {df.shape[0]}\nNumber of columns = {df.shape[1]}')
```

```
Number of rows = 10886
Number of columns = 12
```

Data types of all the attributes

In [ ]:
```
df.dtypes
```

Out[ ]:
```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

conversion of categorical attributes to 'category' (If required)

Converting categorical attributes to the 'category' data type in Python (using pandas) is not compulsory, but it can offer several advantages like ; Memory Efficiency,Performance Improvements,Implicit Ordering,Compatibility

In [ ]:
```
#converting  col(datetime) to standard datetime dtype
df['datetime']=pd.to_datetime(df['datetime'])

#Now change required columns to categorical data type
cols=['season','holiday','workingday','weather']
for col in cols:
  df[col]=df[col].astype('object')
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
datetime      datetime64[ns]
season                object
holiday               object
workingday            object
weather               object
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

statistical summary :

In [ ]:
```
df.iloc[:,1:].describe(include='all')
```

Out[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.
unique	4.0	2.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	
top	4.0	0.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.

missing value detection

In [ ]:

```
df.isnull().any()
```

Out[ ]:

```
datetime      False
season        False
holiday       False
workingday    False
weather       False
temp          False
atemp         False
humidity      False
windspeed     False
casual        False
registered    False
count         False
dtype: bool
```

Insight:

The above mentioned cell gives us the information about all the columns, their data types and Non-Null Count.There is no null values in this dataset.

In [ ]:
```
df.iloc[:,1:].nunique()
```

Out[ ]:
```
season          4
holiday         2
workingday      2
weather         4
temp           49
atemp          60
humidity       89
windspeed      28
casual        309
registered    731
count         822
dtype: int64
```

In [ ]:
```
df.iloc[:,1:].nunique().sum()
```

Out[ ]:
```
2100
```

Insight: The above cell gives us the number of unique values present in the different columns of dataset.There is total 2100 unique values.

In [ ]:
```
df.duplicated()
```

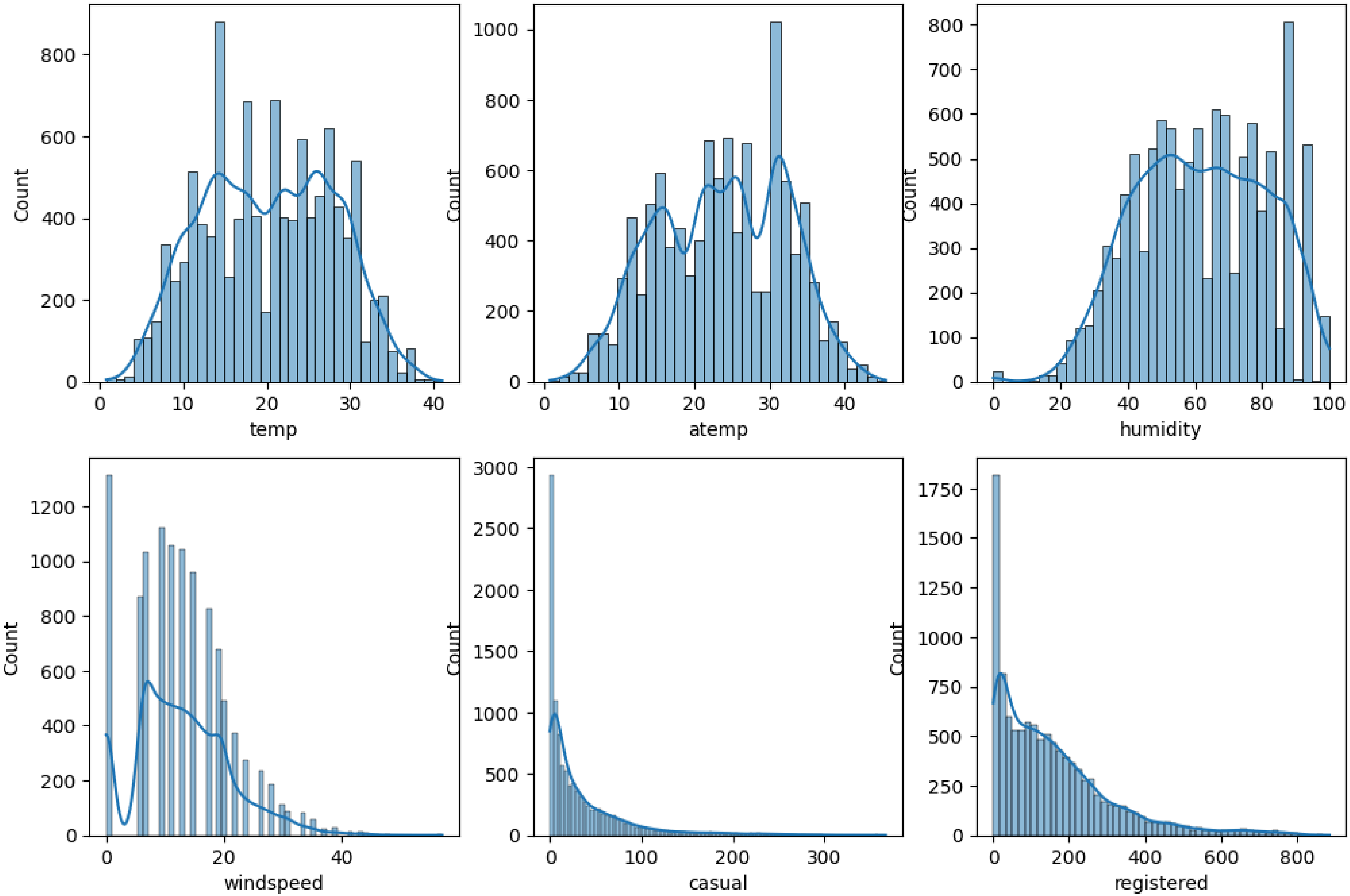
Out[ ]:
```
0        False
1        False
2        False
3        False
4        False
         ...  
10881    False
10882    False
10883    False
10884    False
10885    False
Length: 10886, dtype: bool
```

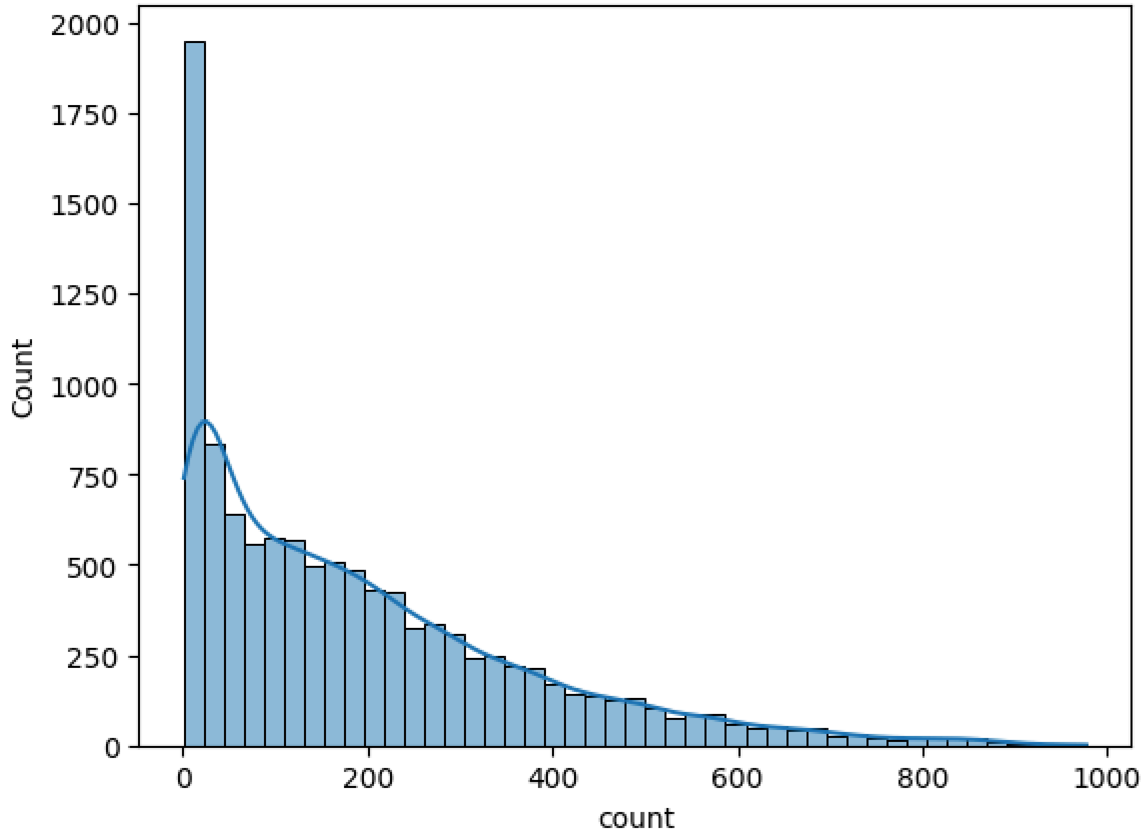
2. Try establishing a Relationship between the Dependent and Independent Variables

Univariate Analysis

Understanding the distribution for numerical variables

In [ ]:
```
cols=['temp','atemp', 'humidity', 'windspeed', 'casual','registered','count']
fig,axis= plt.subplots(nrows=2,ncols=3,figsize=(12,8))
index=0
for row in range(2):
  for col in range(3):
    sns.histplot(df[cols[index]],ax=axis[row,col],kde=True)
    index +=1
plt.show()
sns.histplot(df[cols[-1]], kde=True)
plt.show()
```


Count
temp
atemp
humidity
windspeed
casual
registered



Insights:

1.casual, registered and count somewhat looks like Log Normal Distribution

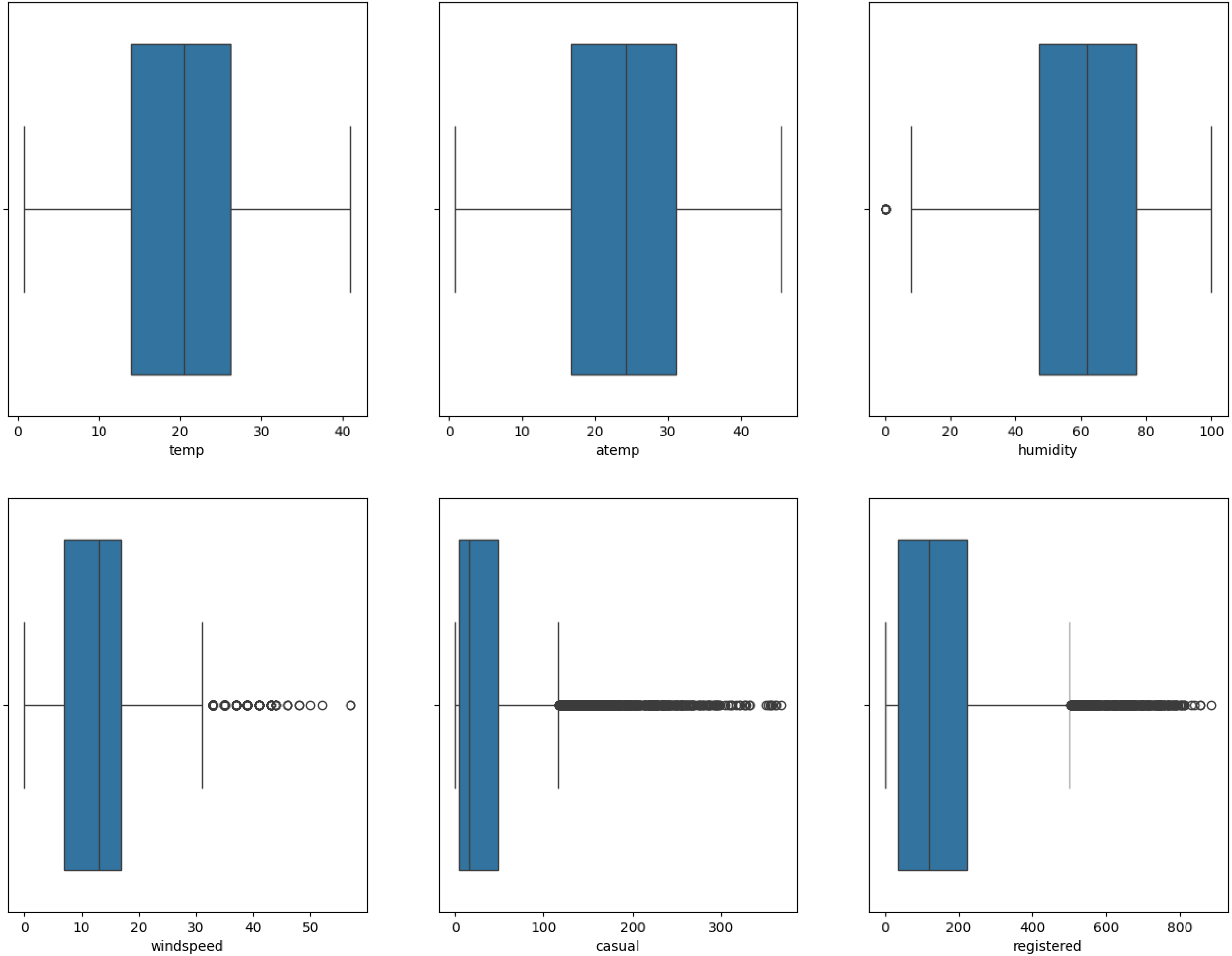
2.temp, atemp and humidity looks like they follows the Normal Distribution.

3.windspeed follows the binomial distribution

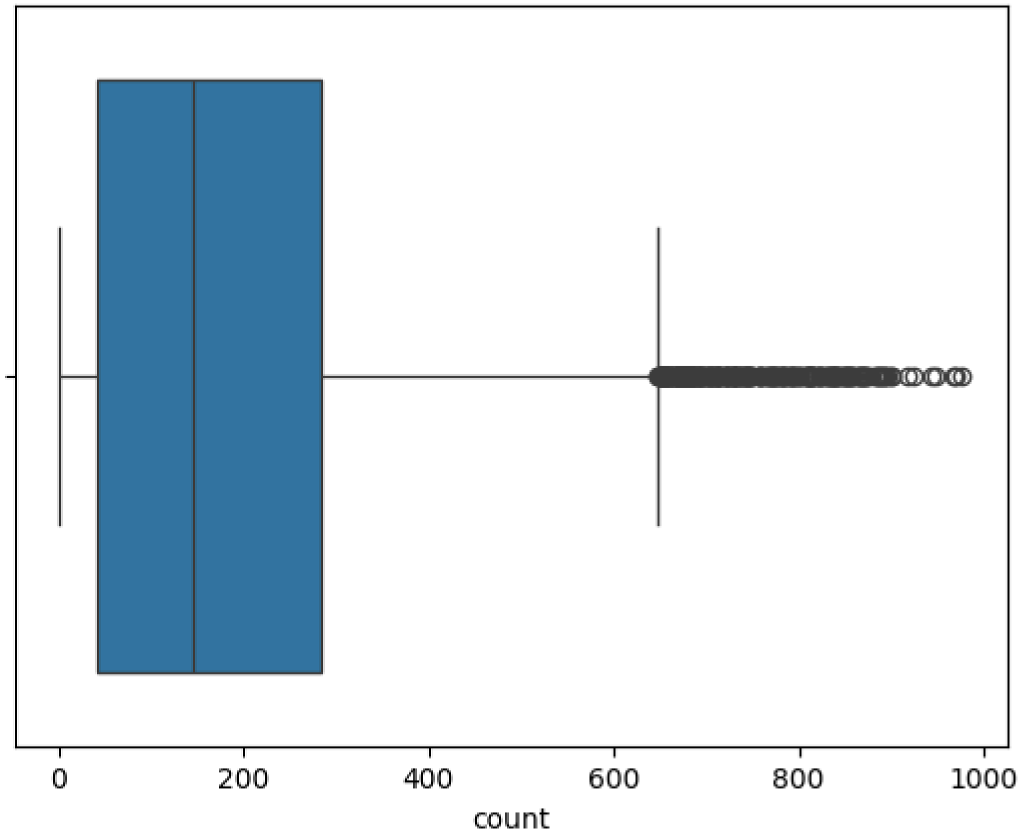
plotting box plots to detect outliers in the data

In [ ]:
```
fig,axis=plt.subplots(nrows=2,ncols=3,figsize=(16,12))
index=0
for row in range(2):
  for col in range(3):
```

```
    sns.boxplot(x=df[cols[index]],ax=axis[row,col])
    index+=1
plt.show()
sns.boxplot(x=df[cols[-1]])
plt.show()
```



0 10 20 30 40
temp
0 10 20 30 40
atemp
0 20 40 60 80 100
humidity
0 10 20 30 40 50
windspeed
0 100 200 300
casual
0 200 400 600 800
registered

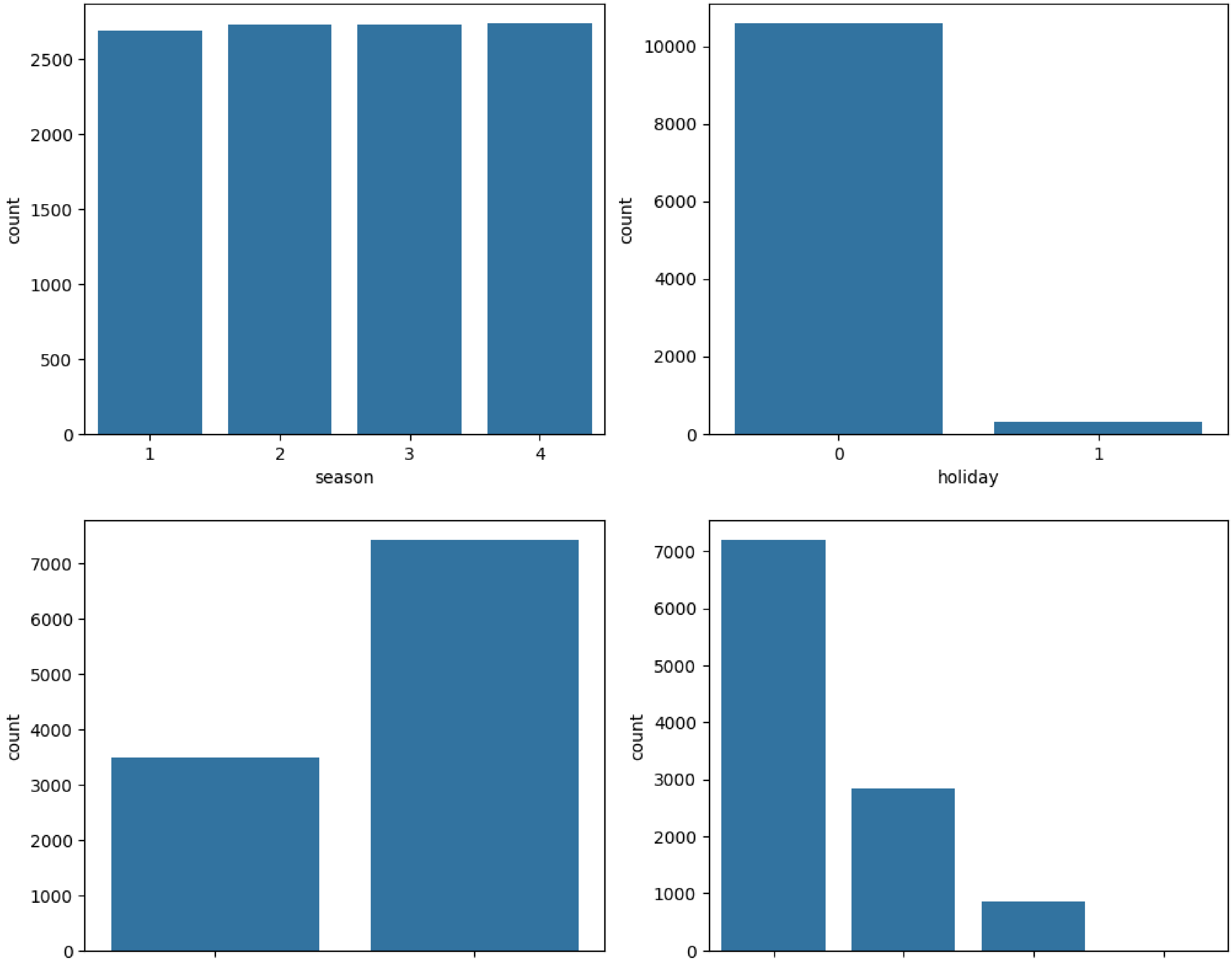


Insights:

Looks like windspeed, casual, registered and count have outliers in the data.

Countplot for categorical columns

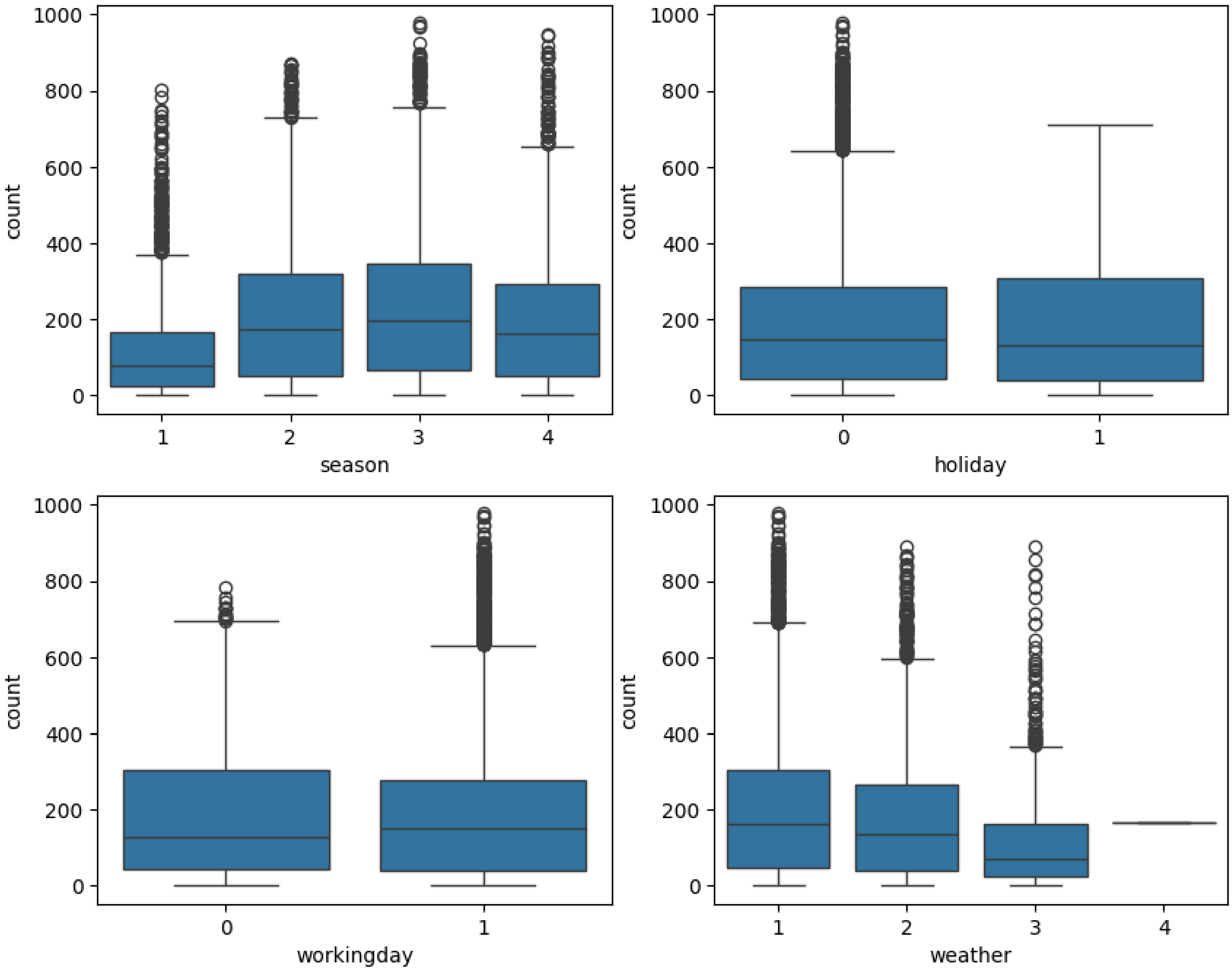
In [ ]:
```
fig,axis=plt.subplots(nrows=2, ncols=2, figsize=(12,10))
cols=['season','holiday','workingday','weather']
index=0
for row in range(2):
  for col in range(2):
    sns.countplot(data=df,x=cols[index],ax=axis[row,col])
    index+=1
plt.show()
```


count
season
1
2
3
4
0
500
1000
1500
2000
2500
count
holiday
0
1
0
2000
4000
6000
8000
10000
count
0
1000
2000
3000
4000
5000
6000
7000
count
0
1000
2000
3000
4000
5000
6000
7000

Bi-variate Analysis

Plotting categorical variables againt count using boxplots

In [ ]:
```
fig,axis=plt.subplots(nrows=2,ncols=2,figsize=(10,8))
cols=['season','holiday','workingday','weather']
index=0
for row in range(2):
  for col in range(2):
    sns.boxplot(data=df,x=cols[index],y='count',ax=axis[row,col])
    index+=1
plt.show()
```


count
season
holiday
workingday
weather
1000
800
600
400
200
0
1
2
3
4

Insights:

1.In summer and fall seasons more bikes are rented as compared to other seasons.

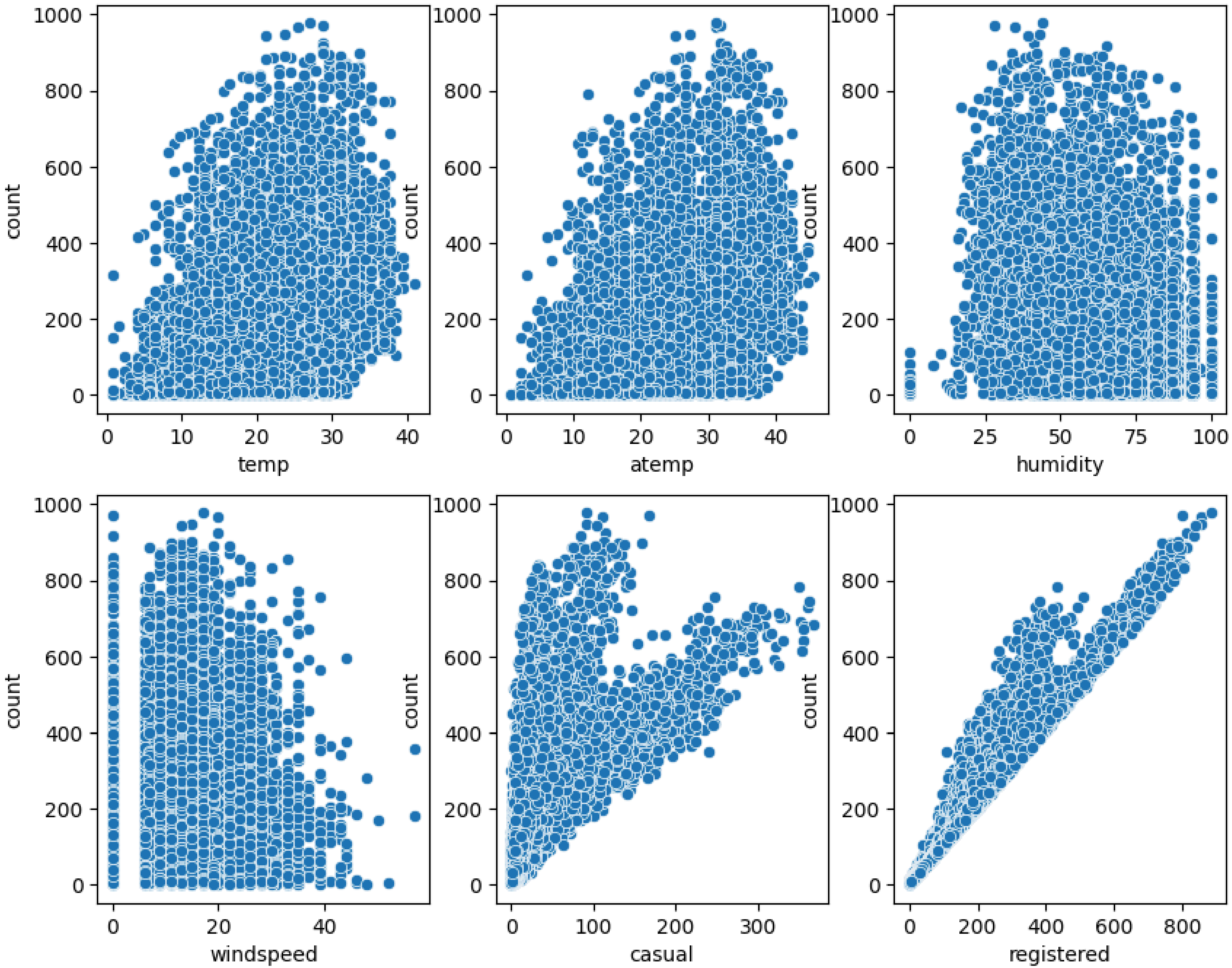
2.Whenever its a holiday more bikes are rented.

3.It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.

4.Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.

Plotting numerical variables againt count using scatterplot

In [ ]:
```
fig,axis=plt.subplots(nrows=2,ncols=3,figsize=(10,8))
cols=['temp','atemp', 'humidity', 'windspeed', 'casual','registered','count']
index=0
for row in range(2):
  for col in range(3):
    sns.scatterplot(data=df,x=df[cols[index]],y='count',ax=axis[row,col])
    index+=1
plt.show()
```



Insights:

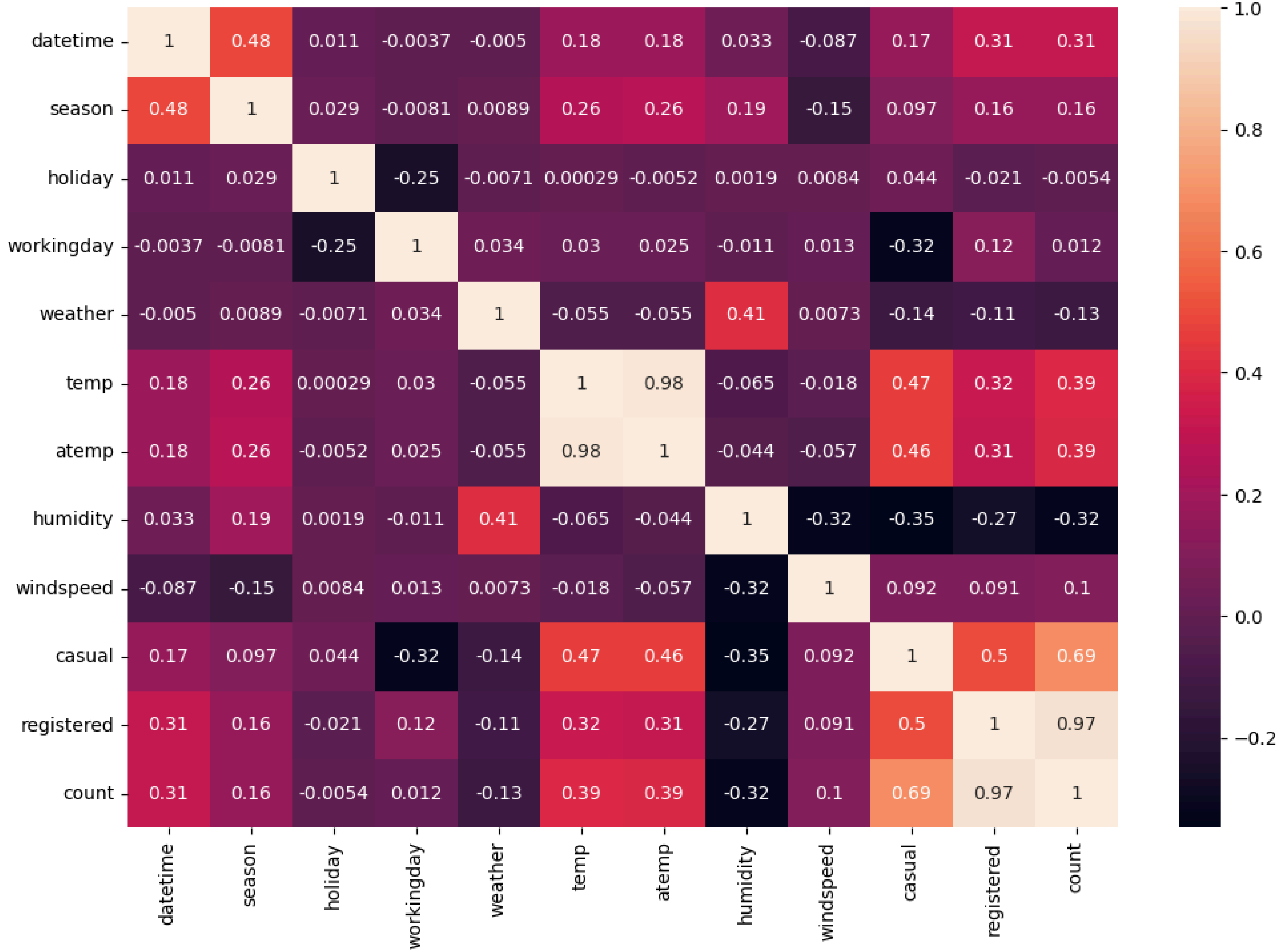
1.Whenever the humidity is less than 20, number of bikes rented is very very low.

2.Whenever the temperature is less than 10, number of bikes rented is less.

3.Whenever the windspeed is greater than 35, number of bikes rented is less.

Understanding the correlation between count and numerical variables

In [ ]:
```
plt.figure(figsize=(12,8))
df.corr()['count']
sns.heatmap(df.corr(),annot=True)
plt.show()
```


datetime
season
holiday
workingday
weather
temp
atemp
humidity
windspeed
casual
registered
count
1 0.48 0.011 -0.0037 -0.005 0.18 0.18 0.033 -0.087 0.17 0.31 0.31
0.48 1 0.029 -0.0081 0.0089 0.26 0.26 0.19 -0.15 0.097 0.16 0.16
0.011 0.029 1 -0.25 -0.0071 0.00029 -0.0052 0.0019 0.0084 0.044 -0.021 -0.0054
-0.0037 -0.0081 -0.25 1 0.034 0.03 0.025 -0.011 0.013 -0.32 0.12 0.012
-0.005 0.0089 -0.0071 0.034 1 -0.055 -0.055 0.41 0.0073 -0.14 -0.11 -0.13
0.18 0.26 0.00029 0.03 -0.055 1 0.98 -0.065 -0.018 0.47 0.32 0.39
0.18 0.26 -0.0052 0.025 -0.055 0.98 1 -0.044 -0.057 0.46 0.31 0.39
0.033 0.19 0.0019 -0.011 0.41 -0.065 -0.044 1 -0.32 -0.35 -0.27 -0.32
-0.087 -0.15 0.0084 0.013 0.0073 -0.018 -0.057 -0.32 1 0.092 0.091 0.1
0.17 0.097 0.044 -0.32 -0.14 0.47 0.46 -0.35 0.092 1 0.5 0.69
0.31 0.16 -0.021 0.12 -0.11 0.32 0.31 -0.27 0.091 0.5 1 0.97
0.31 0.16 -0.0054 0.012 -0.13 0.39 0.39 -0.32 0.1 0.69 0.97 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Insights:

There is a positive corelation between counts and temperature.

There is a negative corelation between counts and humidity

3: Hypothesis Testing:

Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

To check whether there any significant difference between the no. of bike rides on Weekdays and Weekends we will use **2 Sample T Test**

Setting the Hypothesis

Null Hypothesis (H0): No Significant difference between the no. of bike rides on Weekdays and Weekends.

Alternate Hypothesis (H1): Significant difference between the no. of bike rides on Weekdays and Weekends.

Significane Level: **0.05**

We will use the 2-Sample T-Test to test the hypothess defined above

In [ ]:
```
Weekday =df[df['workingday']==1]['count'].values
Weekends=df[df['workingday']==0]['count'].values
```

Before conducting the two-sample T-Test we need to find if the given data groups have the same variance.

In [ ]:
```
print(np.var(Weekday),np.var(Weekends))
```

```
34040.69710674686 30171.346098942427
```

We can say that the variance of two samples are approximately Equal.Hence we can apply **2 SAMPLE T-TEST**

In [ ]:
```
from scipy.stats import ttest_ind

t_stat, p_value = ttest_ind(Weekday, Weekends)
print(f't_stat={t_stat}\np_value={p_value}')
```

```
t_stat=1.2096277376026694
p_value=0.22644804226361348
```

In [ ]:
```
alpha=0.05
if p_value < alpha:
  print('Reject H0:')
  print('There is Significant difference between the no. of bike rides on Weekdays and Weekends')
else:
  print('Fail to reject H0:')
  print('There is Significant difference between the no. of bike rides on Weekdays and Weekends')
```

```
Fail to reject H0:
There is Significant difference between the no. of bike rides on Weekdays and Weekends
```

Conclusions:

1.Whenever its a holiday more bikes are rented.

2.It is also clear from the above test that whenever day is holiday or weekend, slightly more bikes were rented.

Recommendations:

1.Company must provide more offers and discount on the rides in weekdays to increase the number of rider.

2.Rates of rented bikes must be affordable to the customers.

4.Check if the demand of bicycles on rent is the same for different Weather conditions?

To check if the demand of bicycles on rent is the same for different Weather conditions we have to apply **One- Way ANNOVA Test**.

In [ ]:
```
Unique_weather=df['weather'].unique()
Unique_weather
```

Out[ ]: array([1, 2, 3, 4], dtype=object)

Setting the Hypothesis

Null Hypothesis (H0): The demand of bicycles on rent is same for different Weather conditions.

Alternate Hypothesis (H1): The demand of bicycles on rent is different for different Weather conditions.

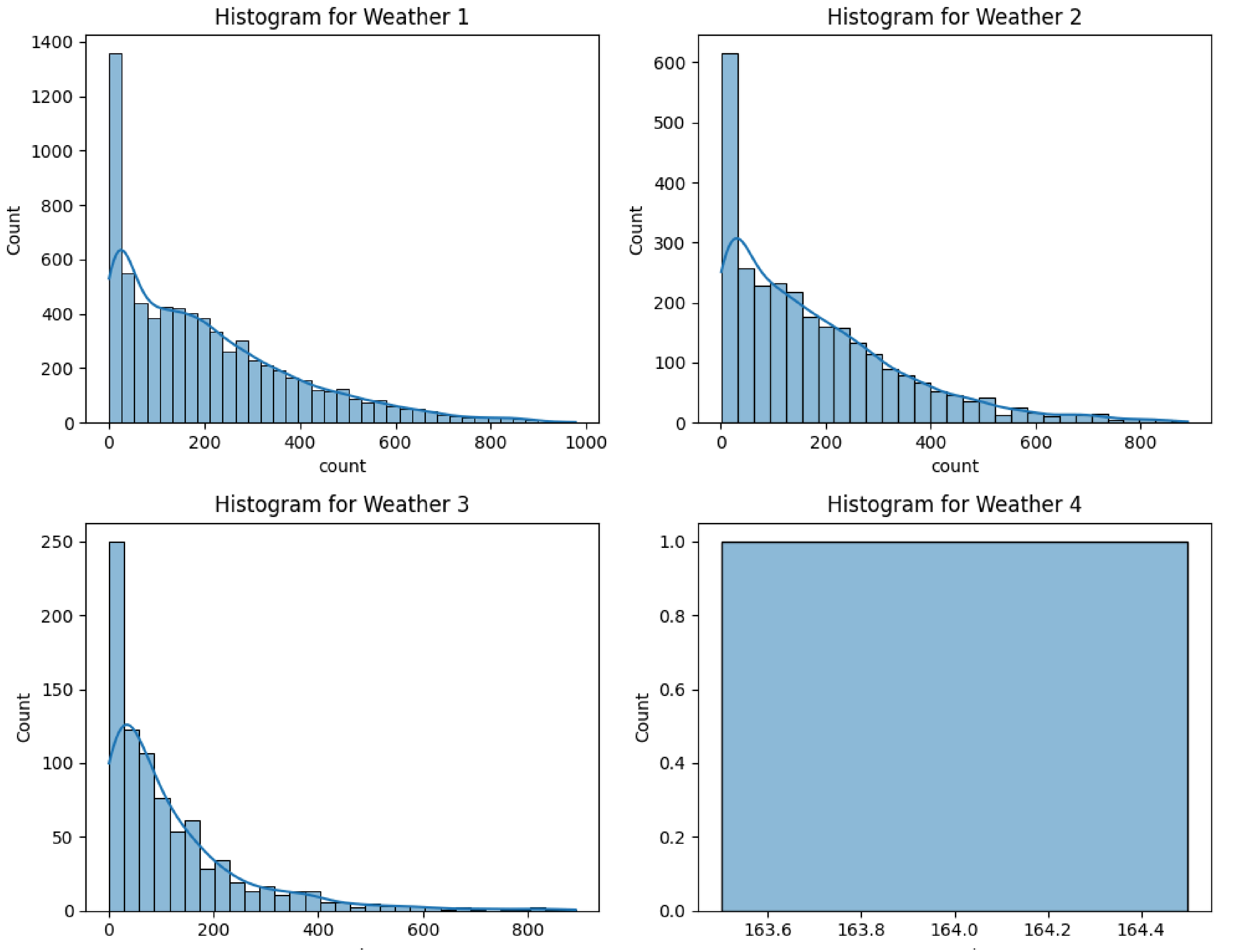
Significane Level: 0.05

Check assumptions of the test

1.Histogram :to check the Normality

In [ ]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Group the data by 'Weather'
groups = df.groupby('weather')['count']

# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# Iterate over the groups and create histograms
index = 1
for row in range(2):
  for col in range(2):

    sns.histplot(groups.get_group(index), ax=axes[row, col], kde=True)
    axes[row, col].set_title(f'Histogram for Weather {index}')
    index += 1

plt.tight_layout()
plt.show()
```



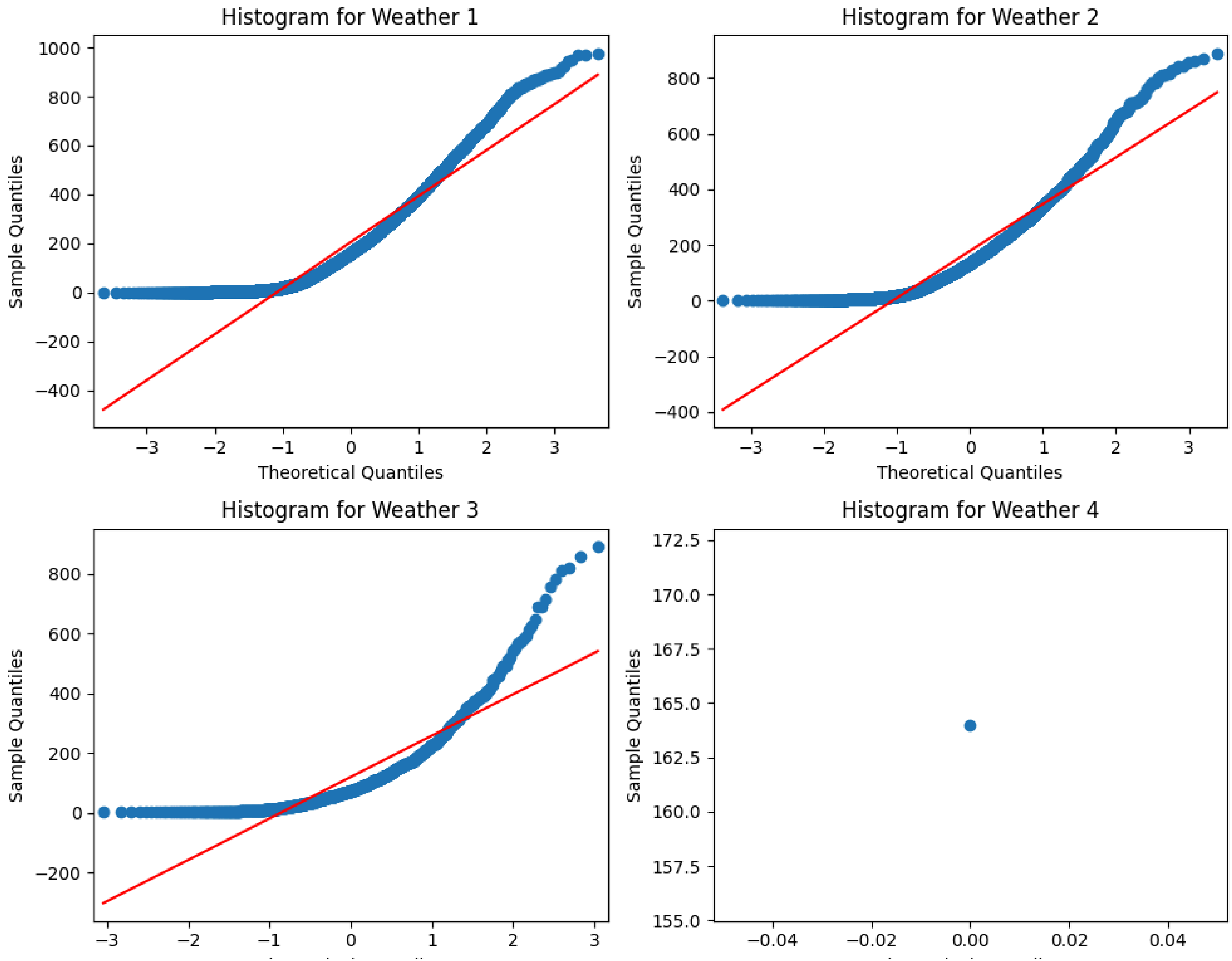
Histogram for Weather 1
Count
count
Histogram for Weather 2
Count
count
Histogram for Weather 3
Count
Histogram for Weather 4
Count

count count

By looking at each graph we can see that they are symmetric.Hence they are not **Bell Curved** or **Normally Distributed** or **Gaussian Distribution**.

In [ ]:
```
from scipy.stats import probplot

from statsmodels.graphics.gofplots import qqplot
```

In [ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import probplot

# Group the data by 'Weather'
groups = df.groupby('weather')['count']

# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# Iterate over the groups and create histograms
index = 1
for row in range(2):
  for col in range(2):

    qqplot(groups.get_group(index), ax=axes[row, col],line="s")
    axes[row, col].set_title(f'Histogram for Weather {index}')
    index += 1

plt.tight_layout()
plt.show()
```



Histogram for Weather 1
Histogram for Weather 2
Histogram for Weather 3
Histogram for Weather 4
Sample Quantiles
Theoretical Quantiles
1000
800
600
400
200
0
−200
−400
−3
−2
−1
0
1
2
3
172.5
170.0
167.5
165.0
162.5
160.0
157.5
155.0
−0.04
−0.02
0.00
0.02
0.04

Theoretical Quantiles Theoretical Quantiles

BY looking at each QQ-Plot we can say that the line is crooked or the dots are scattered,the blue dots does not fall close to **Red Line**.Hence does not follow a **Normal Distribution**

.2.Equality of Variance

In [ ]:
```
from scipy.stats import levene

# Extract weather conditions and corresponding counts
weather_conditions = df['weather'].unique()
data_by_weather = [groups.get_group(condition) for condition in weather_conditions]

# Perform Levene test
statistic, p_value = levene(data_by_weather[0], data_by_weather[1], data_by_weather[2], data_by_weather[3])

# Print the results
print("Levene Test Results:")
print(f"Statistic: {statistic}")
print(f"P-value: {p_value}")
```

```
Levene Test Results:
Statistic: 54.85106195954556
P-value: 3.504937946833238e-35
```

In [ ]:
```
# H0: Variances are equal
# Ha: Variances are not equal
if p_value < 0.05:
    print("Variances are not equal")
else:
  print('Variances are equal')
```

```
Variances are not equal
```

Above provided **LEVENE TEST** clarify that Variance are not Homogenous.

Hence we have to **reject One Way ANNOVA**

So now we will go for another method i.e **Kruskal Wallis**

The Kruskal-Wallis test is a statistical test used to compare the medians of two or more independent groups.

It is an alternative to the one-way ANOVA, making it useful when data is not normally distributed.

Set the Hypothesis:

Null Hypothesis (H0):

There is no significant difference in the median counts of rented bicycles across different weather conditions.

Alternative Hypothesis (H1):

There is a significant difference in the median counts of rented bicycles across at least two weather conditions.

In [ ]:
```
from scipy.stats import kruskal

# Perform Kruskal-Wallis test
statistic, p_value = kruskal(data_by_weather[0], data_by_weather[1], data_by_weather[2], data_by_weather[3])

# Print the results
print("Kruskal-Wallis Test Results:")
print(f"Statistic: {statistic}")
print(f"P-value: {p_value}")
```

```
Kruskal-Wallis Test Results:
Statistic: 205.00216514479087
P-value: 3.501611300708679e-44
```

In [ ]:
```
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant difference in the median counts of rented bicycles across at least two weather cond
else:
    print("Fail to reject H0")
    print("There is no significant difference in the median counts of rented bicycles across at least two weather cond
```

```
Reject H0
There is a significant difference in the median counts of rented bicycles across at least two weather conditions.
```

After all the statistical Testing we can say that that Number of cycles rented is not similar in different weather conditions.

Conclusion:

1.Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.

2.There is a normal demand for renting the bikes when weather is Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist,Light Snow, Light Rain + Thunderstorm + Scattered clouds

3.Whenever the weather is Clear, Few clouds, partly cloudy,there is tremendous increase in renting the bikes.

Recommendation:

1.I will recommend different type of vehicles on rainy or snowy days,so that weather can't affect the customer.

2.For rainy or snowy days ,proper equipments can be provided to customers like: Raincoat,Shade or protection.

5. Check if the demand of bicycles on rent is the same for different Seasons?

To check if the demand of bicycles on rent is the same for different Saeson conditions we have to apply **One- Way ANNOVA Test**.

In [ ]:
```
Unique_season=df['season'].unique()
Unique_season
```

Out[ ]:
```
array([1, 2, 3, 4], dtype=object)
```

Setting the Hypothesis

Null Hypothesis (H0): The demand of bicycles on rent is same for different Season conditions.

Alternate Hypothesis (H1): The demand of bicycles on rent is different for different Season conditions.

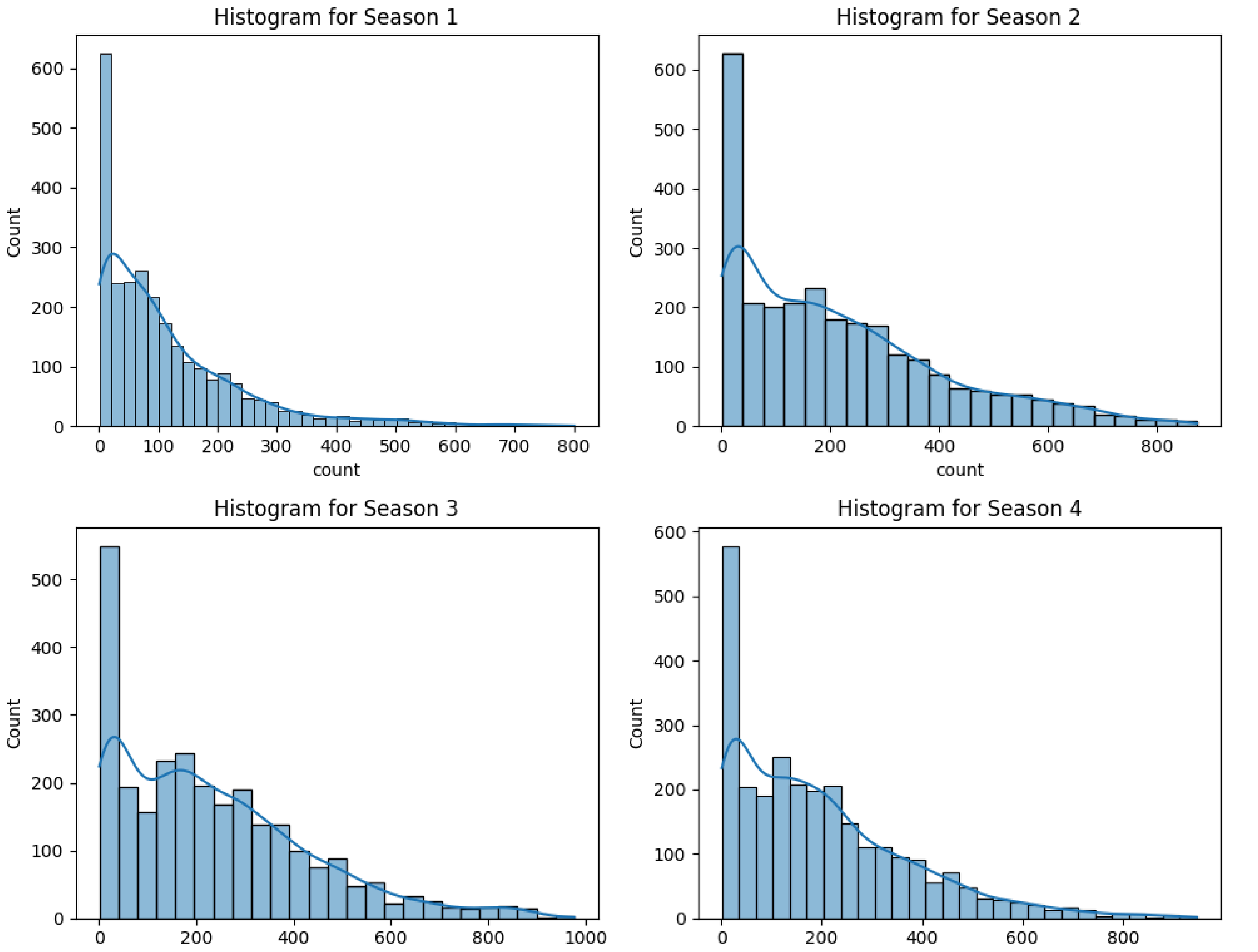
Significane Level: 0.05

Check assumptions of the test

1.Histogram :to check the Normality

In [ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Group the data by 'Season'
groups = df.groupby('season')['count']

# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# Iterate over the groups and create histograms
index = 1
for row in range(2):
  for col in range(2):

    sns.histplot(groups.get_group(index), ax=axes[row, col], kde=True)
    axes[row, col].set_title(f'Histogram for Season {index}')
    index += 1

plt.tight_layout()
plt.show()
```

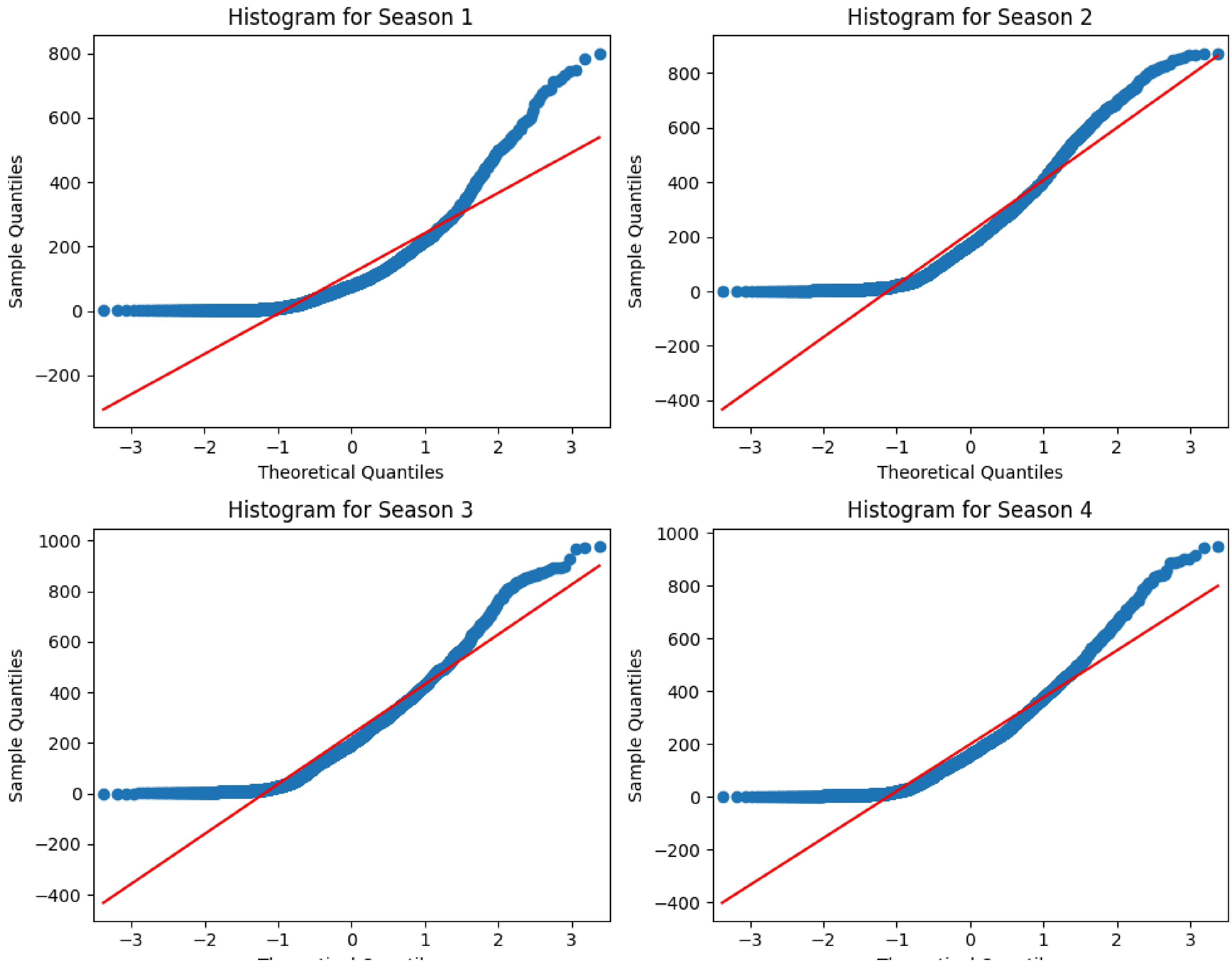


Histogram for Season 1
Histogram for Season 2
Histogram for Season 3
Histogram for Season 4
Count
count

count count

By looking at each graph we can see that they are symmetric.Hence they are not **Bell Curved** or **Normally Distributed** or **Gaussian Distribution**.

In [ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import probplot

# Group the data by 'Season'
groups = df.groupby('season')['count']

# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# Iterate over the groups and create histograms
index = 1
for row in range(2):
  for col in range(2):

    qqplot(groups.get_group(index), ax=axes[row, col],line="s")
    axes[row, col].set_title(f'Histogram for Season {index}')
    index += 1

plt.tight_layout()
plt.show()
```


Histogram for Season 1
800
600
400
200
0
−200
Sample Quantiles
−3 −2 −1 0 1 2 3
Theoretical Quantiles
Histogram for Season 2
800
600
400
200
0
−200
−400
Sample Quantiles
−3 −2 −1 0 1 2 3
Theoretical Quantiles
Histogram for Season 3
1000
800
600
400
200
0
−200
−400
Sample Quantiles
−3 −2 −1 0 1 2 3
Histogram for Season 4
1000
800
600
400
200
0
−200
−400
Sample Quantiles
−3 −2 −1 0 1 2 3

Theoretical Quantiles Theoretical Quantiles

BY looking at each **QQ-Plot** we can say that the line is crooked or the dots are scattered,the blue dots does not fall close to **Red Line**.Hence does not follow a **Normal Distribution.**

Equality of Variance

In [ ]:
```
from scipy.stats import levene

# Extract season conditions and corresponding counts
season_conditions = df['season'].unique()
data_by_season = [groups.get_group(condition) for condition in season_conditions]

# Perform Levene test
statistic, p_value = levene(data_by_season[0], data_by_season[1], data_by_season[2], data_by_season[3])

# Print the results
print("Levene Test Results:")
print(f"Statistic: {statistic}")
print(f"P-value: {p_value}")
```

```
Levene Test Results:
Statistic: 187.7706624026276
P-value: 1.0147116860043298e-118
```

In [ ]:
```
# H0: Variances are equal
# Ha: Variances are not equal
if p_value < 0.05:
    print("Variances are not equal")
else:
  print('Variances are equal')
```

```
Variances are not equal
```

Above provided **LEVENE TEST** clarify that Variance are not Homogenous.

Hence we have to **reject One Way ANNOVA**

So now we will go for another method i.e **Kruskal Wallis**

The Kruskal-Wallis test is a statistical test used to compare the medians of two or more independent groups.

It is an alternative to the one-way ANOVA, making it useful when data is not normally distributed.

Set the Hypothesis:

Null Hypothesis (H0):

There is no significant difference in the median counts of rented bicycles across different season conditions.

Alternative Hypothesis (H1):

There is a significant difference in the median counts of rented bicycles across at least two season conditions.

In [ ]:
```
from scipy.stats import kruskal

# Perform Kruskal-Wallis test
statistic, p_value = kruskal(data_by_season[0], data_by_season[1], data_by_season[2], data_by_season[3])

# Print the results
print("Kruskal-Wallis Test Results:")
print(f"Statistic: {statistic}")
print(f"P-value: {p_value}")
```

```
Kruskal-Wallis Test Results:
Statistic: 699.6668548181988
P-value: 2.479008372608633e-151
```

In [ ]:
```
if p_value < 0.05:
    print("Reject H0")
    print("There is a significant difference in the median counts of rented bicycles across at least two season condit
else:
    print("Fail to reject H0")
    print("There is no significant difference in the median counts of rented bicycles across at least two season cond:
```

```
Reject H0
There is a significant difference in the median counts of rented bicycles across at least two season conditions.
```

After all the statistical Testing we can say that Number of cycles rented is not similar in different weather conditions.

Conclusions:

1.In summer and fall seasons more bikes are rented as compared to other seasons.

2.In Winter and Spring seasons due to fog ,rain ,snow thunderstorm there is decrease in the number of rented bikes.

Recommendations:

1.In summer and fall seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.

2.In Winter and Spring seasons the company have to change the mode of transport for customers on genuine and affordable prices.

6. Check if the Weather conditions are significantly different during different Seasons?

Since both Weather and Seasons are both categorical variables we will use **Chi-Square Test.**

The **chi-square test** is a statistical test used to determine whether there is a significant association between two categorical variables.

Null Hypothesis (H0): Weather is independent of the season.

Alternate Hypothesis (H1): Weather is not independent of the season.

Significance level (alpha): 0.05

In [ ]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[ ]:

weather	1	2	3	4
season				
1	1759	715	211	1
2	1801	708	224	0
3	1930	604	199	0
4	1702	807	225	0

In [ ]:

```
from scipy.stats import chi2_contingency
chi_stat, p_value, df, exp_freq = chi2_contingency(data_table) # chi_stat, p_value, df, expected values
print("chi_stat:",chi_stat)
print("p_value:",p_value)
print("df:",df)
print("exp_freq:",exp_freq)
```

```
chi_stat: 49.158655596893624
p_value: 1.549925073686492e-07
df: 9
exp_freq: [[1.77454639e+03 6.99258130e+02 2.11948742e+02 2.46738931e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80625831e+03 7.11754180e+02 2.15736359e+02 2.51148264e-01]]
```

In [ ]:

```
if p_value <= alpha:
 print("We reject the Null Hypothesis. Meaning that\
 Weather is dependent on the season.")
else:
 print("Since p-value is greater than the alpha 0.05, We do not reject the Null Hypothesis")
```

```
We reject the Null Hypothesis. Meaning that Weather is dependent on the season.
```

After the chi-square Test ,we can say that the Weather conditions are significantly different during different Seasons.

Conclusions:

1.It is concluded that all seasons have their own weather,and are significantly different during different seasons.

2.This have a major impact on the number of rented bikes.In some cases impact is positive and in some cases negative.

Recommendations:

1.I will recommend that the company must provide different vehicles for different seasons and weathers.So that the revenue does not get negatively impacted.

2.The renting prices for diffeent vehicles should not impact the pocket of customer,so that they could not hesitate to rent vehicles other then bike.